A Quantitative Analysis of Developer Information Needs in Software Ecosystems

Nicole Haenni, Mircea Lungu, Niko Schwarz, Oscar Nierstrasz

University of Bern
Switzerland
http://scg.unibe.ch

ABSTRACT

We present the results of an investigation into the nature of information needs of software developers who work in projects that are part of larger ecosystems. This work is based on a quantitative survey of 75 professional software developers. We corroborate the results identified in the survey with needs and motivations proposed in a previous survey and discover that tool support for developers working in an ecosystem context is even more meager than we thought: mailing lists and internet search are the most popular tools developers use to satisfy their ecosystem-related information needs.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; D.2.9 [**Software Engineering**]: Management; H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Human Factors, Management, Measurement

Keywords

Software ecosystems, programmer needs, open source software, program comprehension, frameworks and libraries

1. INTRODUCTION

Open source software is based on transparency and reuse: almost no useful open source software project consists of an isolated island. Instead, every project depends on other projects, frameworks, libraries, platforms, and often other projects depend on it. The beauty of this situation is that while functionally dependent, these projects can maintain complete organizational independence. Drawing lessons from the success of the open source model, large companies have been organizing teams that maintain administrative autonomy while co-evolving software and services.

ECSAW August 25 - 29 2014, Vienna, Austria

http://dx.doi.org/10.1145/2642803.2642815

We use the term *software ecosystem* to refer to a collection of software projects that must co-evolve in a context where they functionally depend on each other while maintaining organizational independence [1]. In an ecosystem, the functional interdependence between projects poses special challenges to both the developers of the dependent project and those of the project that is depended on. However, these challenges have been studied far too little.

In our first qualitative study we investigated what information developers need [2]. In an open-question survey we asked framework and library developers about their needs and current practices. We discovered that open source developers take two different perspectives in an ecosystem context. They take the *upstream* perspective when interacting with projects that rely on their source code, and they take the *downstream* perspective when interacting about the projects they rely on. Each perspective brings different challenges, and addressing these challenges requires specific but divergent information needs.

Our preliminary findings show that downstream needs fall into three categories roughly corresponding to the different stages in their relation with an upstream: selection-, adoption-, and co-evolution-related needs. Upstream needs fall in two categories: project statistics and code usage.

The needs, the motivations behind them, and the current practices that we proposed in our previous, qualitative study, were based on our own analysis and synthesis of the interview data, and thus, might be biased.

To validate the proposed needs, motivations, and practices we conducted a quantitative follow-up study. 75 professional developers rated statements based on our initial findings in a closed-ended questionnaire.

In this paper we report the results of the quantitative survey. What we see is that most of the needs that we elicited in the previous study are confirmed. We also see that the practices that we discovered previously have less support than we expected, with mailing lists and searching the internet being the main tools used by the developers. This indicates opportunities for future research.

2. METHODOLOGY

We use a sequential exploratory design [3, Chapter 3] for our research. It is a mixed research methods strategy that consists of a qualitative investigation followed by a quantitative validation survey.

In our qualitative study [2] we asked the interviewed developers what their information needs were corresponding to their upstream and downstream roles in the software ecosys-

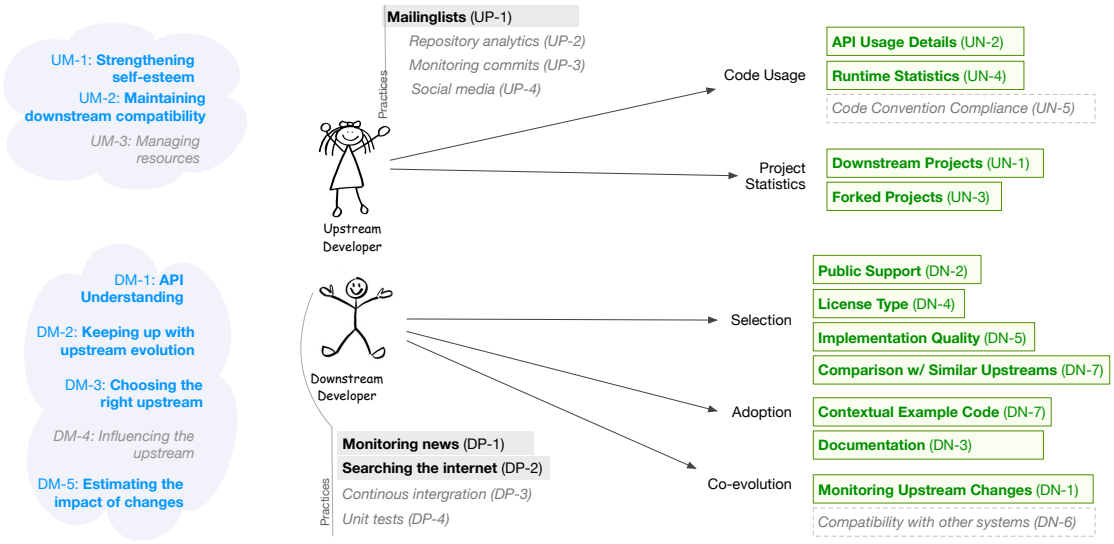


Figure 1: A visual summary of the motivations, information needs and practices identified in our previous, qualitative study. The aspects which are strongly corroborated in the current, quantitative study are in bold.

tem in which they craft software. To analyze the answers we received as free-form text, we applied a grounded theory methodology as introduced by Strauss and Corbin [4]. In the open coding process we identified emerging topics by labeling the text answers line-by-line with appropriate concepts. To ensure quality this procedure was repeated independently by the three mentioned first authors of the paper.

The results of the data analysis phase were lists of information needs, developer motivations, and current practices. Figure 1 synthesizes the main categories we discovered: the motivations on the left, the information needs on the right, and the current practices near the developer figures. Due to lack of space, the figure superimposes information from the current study. Specifically, those items which were strongly corroborated by the current study are in bold, while the others are italic and grayed out.

The qualitative results from our previous study served as an initial position to formulate suitable propositions. With our mixed-methods research strategy we corroborate the importance of the reported information needs and test their consistency by a quantitative investigation. To validate these statements we conducted a quantitative closed-questioned online survey[1]. This time, the participants were randomly selected from mailing list subscriptions. They did not answer with free text, but instead their answers ranged from full disagreement to full agreement on a series of numerical 5-point Likert items.

A total of 51 Likert item questions were asked; 26 questions (Q1.1–Q3.5) to upstream developers and 25 questions (Q4.1-Q6.7) to downstream developers. Furthermore, we asked several pre-survey questions about developer background and three voluntary open-ended post-survey ones. Where applicable, we include quotations from the participants. These quotations are labelled with codes of the form $LS - NN$ where NN is the number of the participant.

To reach a random sample, we advertised the survey in various mailing lists including: Open JDK, Processing.js, jQuery, SciPy, NumPy, Pharo, Squeak, Seaside, Drupal, Core-audio, Apache Hadoop, Apache Cassandra, Google WebToolkit, Ubuntu, Soot and Zend Framework.

We received 75 responses, 46 from upstream developers, and 29 from downstream developers. Even though no response rate could be determined, we reached participants across the world (46% from Europe, 32% North America, 8% from Asian, 6% from South America, 4% from Australia and 1% from Africa).

The distribution of practical knowledge of the respondents is shown in Table 1. Almost half of them have more than ten years and less than a fifth stated to have less than five years experience.

Professional Experiences	Respondents [%]
$<$ 5 years	17
5 - 10 years	29
11 - 20 years	22
$>$ 20 years	22
no answers	10

Table 1: Distribution of respondents' experience

3. UPSTREAM DEVELOPER NEEDS

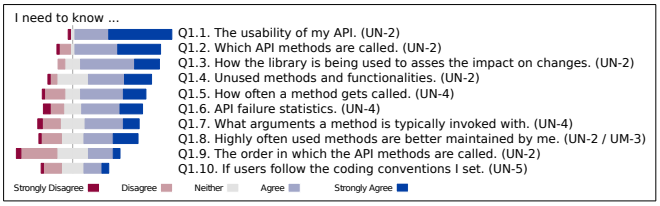
We organize the needs in subsections. At the begining of every subsection we provide a graphic which summarizes the agreement our 75 respondents showed to the questions that pertain to the needs in that section. The data behind the charts is available online[2]. Where appropriate, in the text, we highlight key findings in a box. A green checkmark and a red one mark respectively a need that is corroborated by the current study and one that is not.

[1]The survey is available at http://goo.gl/q2ABRd

[2]http://scg.unibe.ch/research/ecosystem-needs

3.1 Needs Related to Code Usage

These developer needs detail how people use source code[3].



The information need with the highest confirmation rate was "API usage details" (UN-2)[4]. The most agreed upon statement for developers was that they want to know the usability of their API. The next was more detailed: what methods are called by users followed by what methods are not used.

✓ *API usage details* (UN-2)

(Q1.1 – Q1.3) More than 90% either agree or strongly agree that the usability of an API is an important aspect. Just as many people agree as strongly agree in being interested in what API methods are called and how their library is used. To assess the impact of changes, 77.7% of participants want to know how others use their library. A respondent reasons that this is *"not just about minimizing the impact of changes, but also about seeing what's awkward, what features are used in conjunction and which independently, which areas are performance sensitive etc" (LS-57).*

(Q1.9) The importance of "Runtime statistics" (UN-4) is less supported by developer feedback. Although originally it seemed like a legitimate need, 69.9% do not care how downstream developers apply the order of method calls.

✓ *Runtime statistics* (UN-4)

Our findings corroborate the importance of this need.

(Q1.6) 61% agreed or strongly agreed that they want to know API failure statistics.

(Q1.5 and Q1.7) 63.0% agreed or strongly agreed to needing to know the number of method calls ; 53% want to know what parameters a method needs.

✗ *Code convention compliance* (UN-5)

The least agreed upon need is code convention compliance.

(Q1.10) Only a slight majority of developers care if their coding conventions are respected by the downstream. A remaining two thirds strongly or simply disagree.

As these results are inconclusive we consider this need as not being corroborated by the current study.

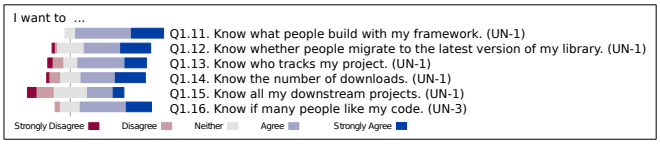
Upstream developers share a strong agreement of needing to know how other use their API since they want to improve their own code.

[3]The numbering of the needs is in sync with the numbering in our previous qualitative analysis report

[4]We asked more questions about API usage since our initial quantitative survey indicated that this need arise five times more than other needs

3.2 Needs Related to Project Statistics

The responses to our questions on project statistics-related information needs can be seen bellow.



The information need "Downstream projects" (UN-1) received strongly positive feedback. The most-agreed-upon statement was that developers need to know what clients build with their code.

✓ *Downstream projects* (UN-1)

We asked developers about information needs with respect to their clients.

(Q1.11 – Q1.14) Two thirds of the participants want to know who is tracking their current project (69.5%, Q1.13) and are interested in the number of downloads (65.2%, Q1.14). Almost 90% agree or strongly agree that they want to know what people built with their provided framework or library (Q1.11). Two-thirds are interested in knowing whether the downstream is migrating to the latest version (Q1.12).

(Q1.15) Two thirds are undecided, disagree or strongly disagree that they need to know all their downstream projects. This statement has the most undecided votes in this category. This is important information for future tool builders since tracking the complete downstream would be clearly impossible.

✓ *Forked projects* (UN-3)

This category is about cloned source code bases. This need is strongly supported.

(Q1.16) 73.9% are interested whether people like their code. When people can not influence the upstream code, they might clone the code base for their own purposes. A majority of the upstream developers confirm that they do not track any forks as the statement Q3.5 indicates.

Upstream developers want to know more than just the number or downloads, followers *etc.*. A comparison across the statements indicates the need of knowing details about how downstream uses their code.

3.3 Discussion of Upstream Findings

Our results show that upstream developers are interested in "API usage details". Researchers are already investigating the automatic analysis of distributed and large-scale repositories: API deprecation in the Smalltalk ecosystem [5], library usage in the Apache ecosystem [6], API evolution in the Android ecosystem [7], code clone detection across projects [8] and ecosystem visualization [9]. However, none of the previous approaches fully addresses the upstream needs reported in this section.

The statement Q1.15 has little support. This is surprising and contradictory to Q1.11. We assume that upstream developers are not interested in the individual project but in the overall usage.

4. DOWNSTREAM DEVELOPER NEEDS

This section discusses the downstream information needs.

4.1 Needs Related to Selection

An overview of the responses to our questions on selection-related information needs can be seen bellow.

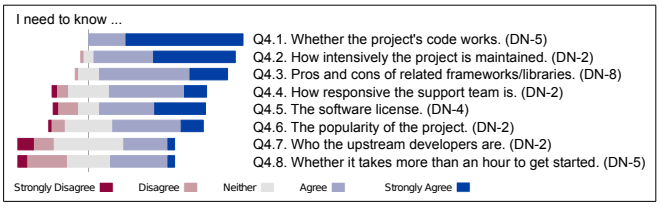


Figure 2: Selection.

✓ *Implementation quality* (DN-5)

(Q4.1) The statement Q4.1 is the strongest statement of this survey. 75.7% strongly agree and 24.3% agree on the need to know whether the project's code works.

(Q4.8) A third of the developers confirm they will dismiss an unfamiliar framework and library if they are unable to make it work within an hour. Two-thirds would spend more than one hour before giving up.

✓ *Available public support* (DN-2)

The finding indicates some degree of inconclusiveness. Results range from strong to low support.

(Q4.2) The second strongest statement is about the intensity of maintenance. 91.4% strongly agree or just agree with this need. A participant confirms: *"As a developer (and user in certain cases), I want to be certain that the community is friendly, accepts [newbies] and responds fast" (LS-48).*

(Q4.7) Two-thirds are not interested in who the developers are. This finding is supported by the motivational statement Q5.4 about trust. In addition, it has a large proportion of undecided answers (45.0%).

When choosing an upstream, developers are less interested in the identity of the developers and more in alternative projects, code quality and the maintenance level.

✓ *Comparsion with similar upstreams* (DN-8)

This information need is strongly supported.

(Q4.3) 84% strongly agree or agree to needing to compare related projects with similar functionalities.

✓ *Licence type* (DN-4)

This information need is strongly supported.

(Q4.5) The selection of a software project depends for 69.6% on its license type.

4.2 Needs Related to Adoption

Adoption is the process of migrating to a new upstream.

In this context, the information needs "Documentation" (DN-3) and "Real contextual sample code" (DN-7) are strongly supported. None of the respondents disagreed with either of the two statements regarding code examples and API documentation.

✓ *Documentation* (DN-3)

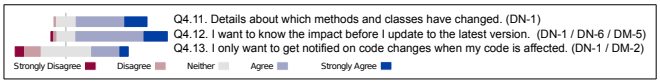
(Q4.10) There is a 90% agreement (and strong agreement) that a good API, design documentation and code examples are essential. One respondent emphasizes: *"[...] this depends on the documentation and ease of use: some frameworks are so easy to use you barely need to read a quick-start document, others are very difficult to learn — sometimes this is because of an over-complicated API, other times its because the concepts are complicated" (LS-42).*

✓ *Real contextual sample code* (DN-7)

(Q4.9) 85% would appreciate extracted code examples illustrating the functionalites provided by their upstream project.

Developers recognize that real code examples from other projects help to understand upstream functionality

4.3 Needs Related to Co-evolution



Developers are interested in details about which methods and classes have changed and whether these changes have an impact on their own source code.

✓ *Monitoring upstream changes* (DN-1)

Evolving a downstream project requires detailed information about source code changes. These include both general bug fixes and release changes.

(Q4.11 and Q4.12) Both reveal a strong agreement on an instant information mechanism.

(Q4.13) There is one Likert item that stands out because of the prevalence of undecided answers. This indicates either a badly-phrased statement or participant lack of experience with notification systems.

Downstream developers want to know the impact *before updating* to the latest version. This includes monitoring the upstream evolution and preview information of changes in implementation details.

4.4 Discussion of Downstream Findings

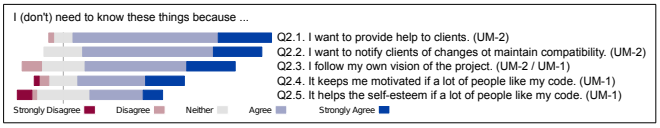
The results show that upstream identity is not relevant to developers. This is the opposite of what developers working in a closed-source context report [10].

In an empirical study on library updates with a collection of Java open-source programs, Dietrich *et al.* show that current practices do not reveal potential impacts when using a newer API version [11]. A solution to this problem and similar ones that can be automatically applied to large-scale software ecosystem development is needed and has the promise of adoption by developers.

5. DEVELOPER MOTIVATIONS

5.1 Upstream Motivations

For the upstream, the motivation "Strengthening self-esteem" is slightly less supported than "Maintaining downstream compatibility", though both are strongly supported.



Strengthening self-esteem (UM-1)

(Q2.4 and Q2.5) Over 70% strongly agree or agree that positive feedback helps them stay motivated. On the other hand, self-esteem was not considered important.

Maintaining downstream compatibility (UM-2)

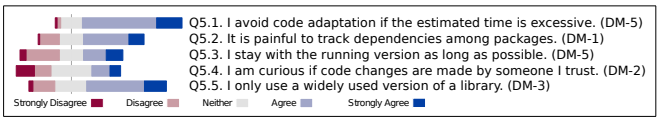
(Q2.1 and Q2.2) There is a strong need to provide help to downstream developers, to notify about code changes and to minimize impacts. A respondent states *"if people are making downstream fixes it would be helpful to know this so that [these changes]can be merged" (LS-42).*

Upstream developers are usually good citizens interested in providing support to downstream.

5.2 Downstream Motivations

Our identified motivations "Choosing the right upstream" (DM-3), "API understanding" (DM-1), "Keeping up with upstream evolution" (DM-2) are all inconclusive.

Motivation "Estimating the impact of changes" is supported. Developers agree that they avoid code migration if the estimated time is excessive.



API understanding (DM-1)

(Q5.2) Difficulties occur to keep an overall overview among code dependencies.

Keeping up with upstream evolution (DM-2)

(Q5.4) A majority of over 70% does not care if code changes are done by developers they trust. Implementation quality is more important than the reputation of the developers.

Choosing the right upstream (DM-3)

(Q5.5) More than half of the respondents have confirmed that the more often a library or framework is already in use the more likely developers will use it, too. This indicates that developers think that frequency of usage may provide information about quality, available support and popularity.

Downstream developers tend to rely on the wisdom of the crowds. They trust more popular libraries.

Estimating the impact of changes (DM-5)

Downstream developers are willing to upgrade to the newest version if adopting the new upstream version is feasible in a reasonable time.

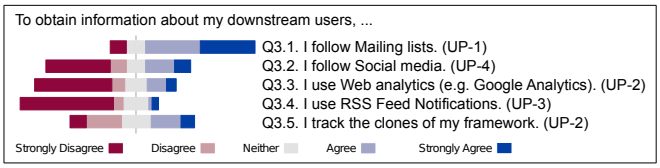
(Q5.1) 85% agreed or strongly agreed that the decision of whether to adapt code or not depends on the estimated time. *(Estimating the impact of changes)*

(Q5.3) One out of four developers would stay with a running version as long as possible. This shows that there is the willingness to keep up with the latest version and at the same time that developers find upgrading a burden.

The decision whether to adapt code or not depends on the estimated time. It is likely that the more complex a software project gets, the more likely it is that developers will omit if possible adaptations such as version updates.

6. DEVELOPER PRACTICES

The statement that got the strongest support from the upstream was that developers follow mailing lists in order to learn about the way their code is used. The other statements received generally negative feedback, indicating either that we overlooked existing tools and practice, or that such tools do not exist.



For the upstream, three practices got strong support: "Monitoring news" (DP-1) — developers subscribe to mailing lists; "Searching the Internet" (DP-2) — developers routinely search for blog posts and tutorials; and "Continuous integration" (DP-3) — developers routinely run integration tests.

"Unit tests" (DP-4) as a practice is inconclusive.

7. THREATS TO VALIDITY

Our questionnaire does not use balanced keying, and therefore is subject to acquiescence bias.

The results of our study depend on the selected participants. Our test population was convenience sampled, the generalizability of our study is limited.

The quantitative approach derives questions that are related to the quantitative approach but there are times when there is only one question per one need. We settled for this approach since we wanted to avoid overloading our respondents with too many questions. This however, limits the generality of our results with respect to the needs.

8. RELATED WORK

Ko *et al.* [12] conducted a study in finding information needs in development teams. Their findings include 21 different types of information in seven categories. The majority refer to knowledge of software artifacts or co-workers.

Seichter *et al.* [13] examine an information retrieval management system for software artifacts to improve collaboration. They define types of interactions but do not declare specific information needs.

Begel *et al.* [10] asked programmers inside Microsoft company about inter-team collaboration problems. They identified and grouped 31 information needs into eight categories.

Phillips *et al.* [14] identify information needs to integrate branched version of a software project. They found four needs: Identifying conflicts before they arise, monitoring features with their dependencies, tracking measured data about number of bugs, test results *etc.*

Jansen reports that the choices that developers make are not always technical, but they are also business related [15].

9. CONCLUSION

Our findings after interviewing a number of 75 developers corroborate almost all the needs discovered in our previous, qualitative study. We conclunde that there is a lack of adequate tool support for developers working in an ecosystem context. We believe that at the intersection of strong information needs, inappropriate practices, and a new research field, lays great potential for future impactful research.

Acknowledgments. We thank Andrei Chis for valuable feedback on this paper. We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "Agile Software Assessment" (SNSF project No. 200020-144126/1, Jan 1, 2013 - Dec. 30, 2015).

10. REFERENCES

[1] Mircea Lungu. *Reverse Engineering Software Ecosystems*. PhD thesis, University of Lugano, November 2009.

[2] Nicole Haenni, Mircea Lungu, Niko Schwarz, and Oscar Nierstrasz. Categorizing developer information needs in software ecosystems. In *Proceedings of the 1st Workshop on Ecosystem Architectures*, pages 1–5, 2013.

[3] John W Creswell and Vicki L Plano Clark. *Designing and conducting mixed methods research*. Wiley Online Library, 2007.

[4] Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications Inc., 1998.

[5] Romain Robbes, Mircea Lungu, and David Röthlisberger. How do developers react to api deprecation?: the case of a smalltalk ecosystem. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, page 56. ACM, 2012.

[6] Yana Momchilova Mileva, Valentin Dallmeier, Martin Burger, and Andreas Zeller. Mining trends of library usage. In *Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops*, IWPSE-Evol '09, pages 57–62, New York, NY, USA, 2009. ACM.

[7] Tyler McDonnell, Baishakhi Ray, and Miryung Kim. An empirical study of api stability and adoption in the android ecosystem. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 70–79. IEEE, 2013.

[8] Niko Schwarz, Mircea Lungu, and Romain Robbes. On how often code is cloned across repositories. In *Proceedings of the 2012 International Conference on Software Engineering*, ICSE 2012, pages 1289–1292, Piscataway, NJ, USA, 2012. IEEE Press.

[9] Mircea Lungu, Michele Lanza, Tudor Gîrba, and Romain Robbes. The Small Project Observatory: Visualizing software ecosystems. *Science of Computer Programming, Elsevier*, 75(4):264–275, April 2010.

[10] Andrew Begel, Yit Phang Khoo, and Thomas Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 125–134, New York, NY, USA, 2010. ACM.

[11] Jens Dietrich, Kamil Jezek, and Premek Brada. Broken promises: An empirical study into evolution problems in java programs caused by library upgrades. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, pages 64–73. IEEE, 2014.

[12] Andrew J. Ko, Robert DeLine, and Gina Venolia. Information needs in collocated software development teams. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 344–353, Washington, DC, USA, 2007. IEEE Computer Society.

[13] Dominik Seichter, Deepak Dhungana, Andreas Pleuss, and Benedikt Hauptmann. Knowledge management in software ecosystems: software artefacts as first-class citizens. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, pages 119–126. ACM, 2010.

[14] Shaun Phillips, Guenther Ruhe, and Jonathan Sillito. Information needs for integration decisions in the release process of large-scale parallel development. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1371–1380. ACM, 2012.

[15] Slinger Jansen. How quality attributes of platform architectures influence software ecosystems. In *Proceedings of the 1st Workshop on Ecosystem Architectures*, 2013.